Robert Stanik śr 15:15 - 16:55 235508 Termin do 28.03.2018

Opis i wprowadzenie

W projekcie zostały zaimplementowane trzy algorytmy sortowania, są to: sortowanie szybkie(*quicksort*), sortowanie przez scalanie(*merge sort*) oraz sortowanie introspektywne (*introspective sort*). Implementacja oparta jest na materiałach dostępnych w internecie. W celu poprawnej kompilacji programu trzeba użyć standardu -stdc++11. Programy zostały zrealizowane zgodnie z instrukcją. W celu weryfikacji poprawności sortowania program został zmieniony w sposób taki że elementy tablicy zostały wyświetlone i sprawdzone w sposób wizualny, oraz powstała funkcja która sprawdza poprawność sortowania.

Sortowanie szybkie został zaimplementowany metodą ograniczającą rekursje, dzięki temu została ograniczona złożoność pamięciowa z O(n) do O(log n), przez co algorytm działa szybciej. Algorytm działa na zasadzie „dziel i zwyciężaj”. Średnia złożoność obliczeniowa wynosi O(nlog n), a pesymistyczna $O(n^2)$. W celu lepszego uświadomienia sobie jak ważny jest tzw pivot, zostały wykonane dwa badania, innym czynnikiem było też ułatwienia analizy sortowania introspektywnego

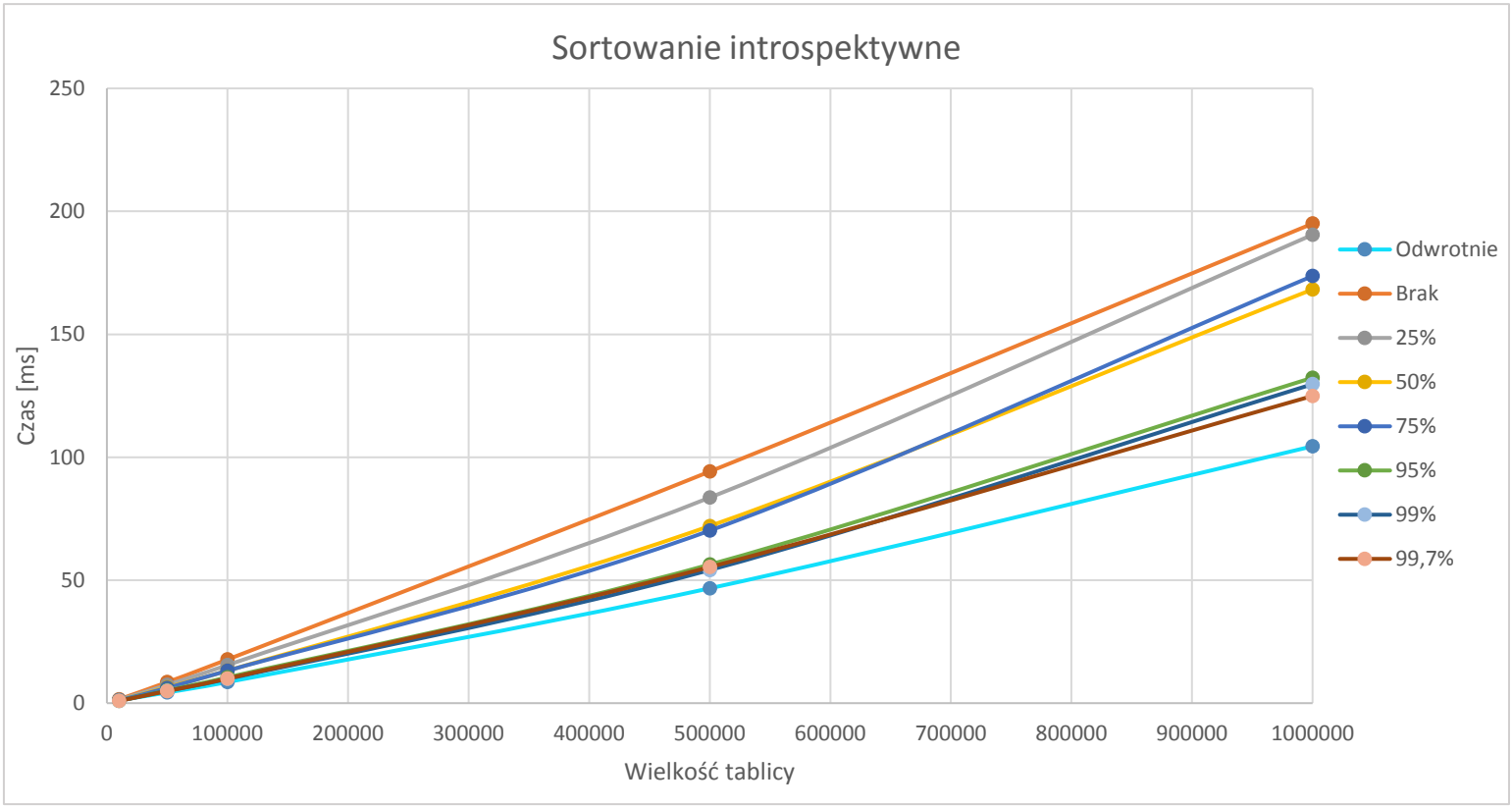
Sortowanie przez scalanie podobnie jak sortownie szybkie działa na zasadzie „dziel i zwyciężaj”. Posiada złożoność obliczeniową O(nlog n), i to niezależnie od ułożenia elementów na tablicy. Złożoność pamięciowa wynosi O(n).

Sortowanie introspektywne jest złożona z dwóch sortowań. Wykorzystuje dzielenie tablicy na dwie części z sortowania szybkiego, oraz inny algorytm sortowania z złożonością obliczeniową O(nlog n), w tym przypadku sortowanie przez kopcowanie. Taki zabieg powoduje że ilość rekurencji potrzebnych w najgorszym przypadku sortowania szybkiego zostaje wyeliminowany. Dzięki temu złożoność obliczeniowa jest równa O(nlog n).

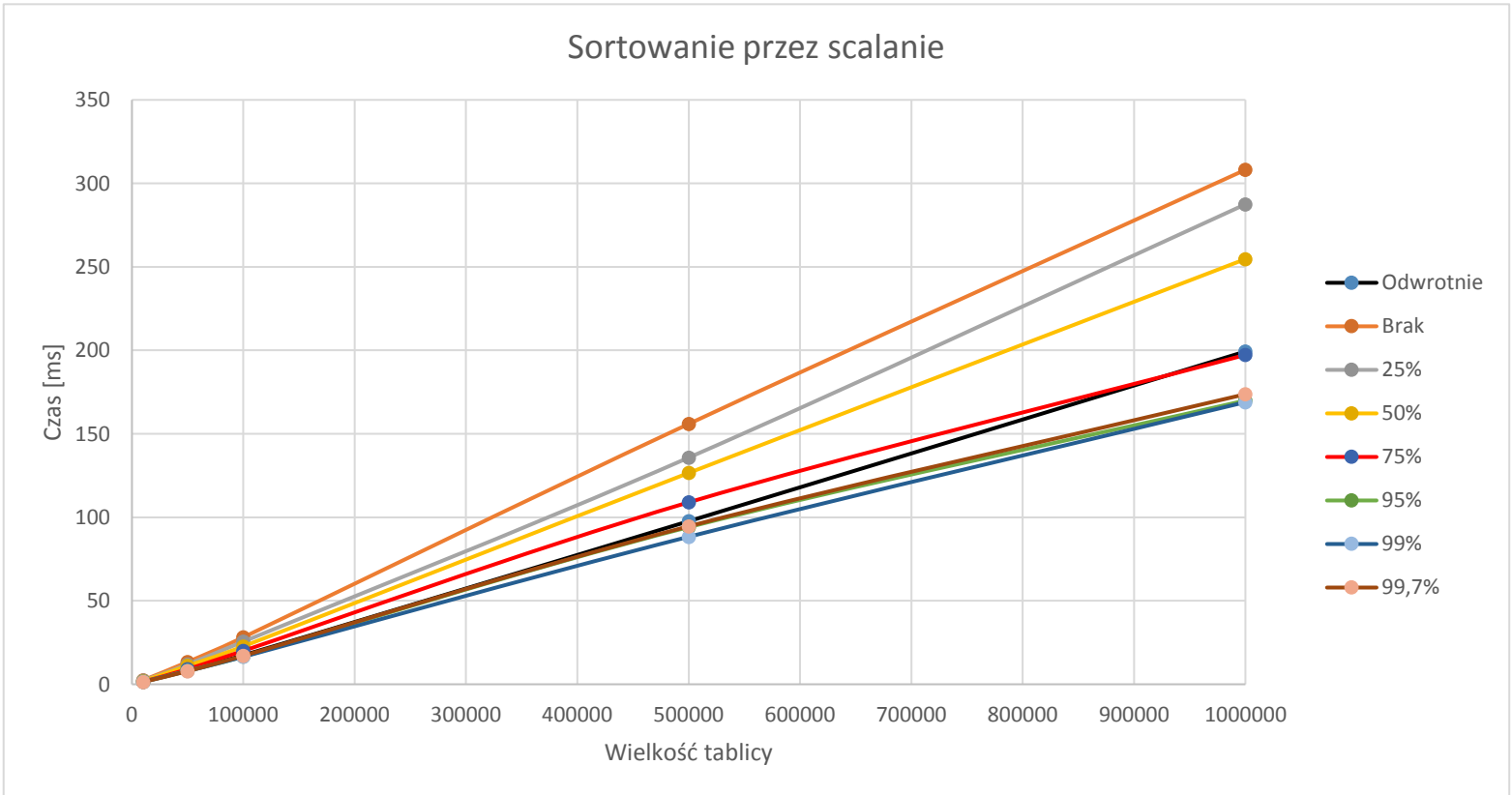
Przebieg eksperymentu

Po wcześniejszym sprawdzeniu poprawności sortowania dla każdego algorytmu, program został uruchomiony i na wyjściu wypisywał poszczególne czasy dla każdych kombinacji. Jednorazowe uruchomienie programu powoduje wypisanie wszystkich możliwości tzn. każda wielkość tablicy i dla niej poszczególne wytyczne zgodne z instrukcją, w jednym przypadku ilość powtórzeń eksperymentu został zredukowany do 10 razy ponieważ czas sortowania był bardzo długi.

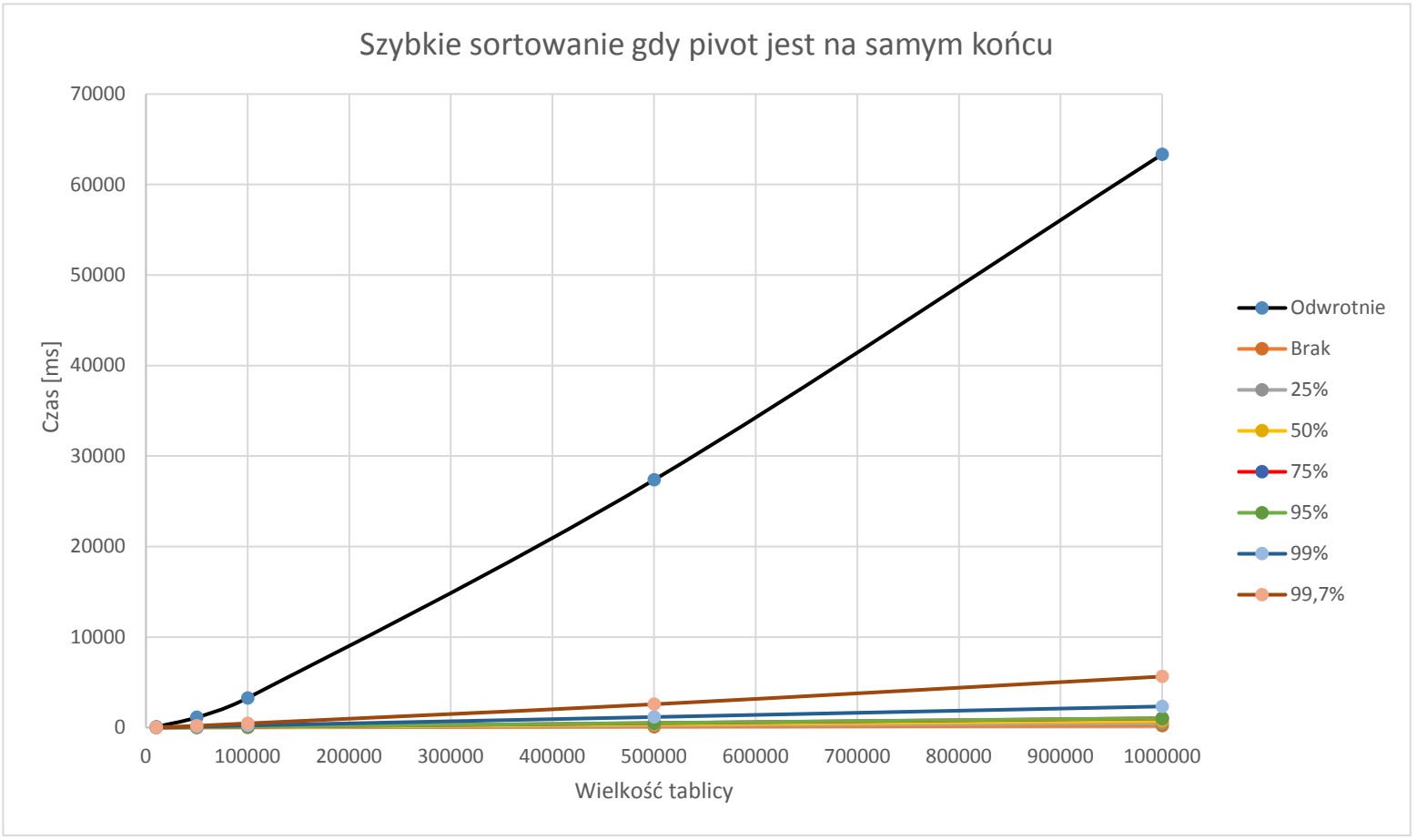
Wykresy zostały zrobione na podstawie danych uzyskanych dzięki badaniom z wykorzystaniem napisanemu programowi i przedstawiają zależność czasu od wielkości tablicy.



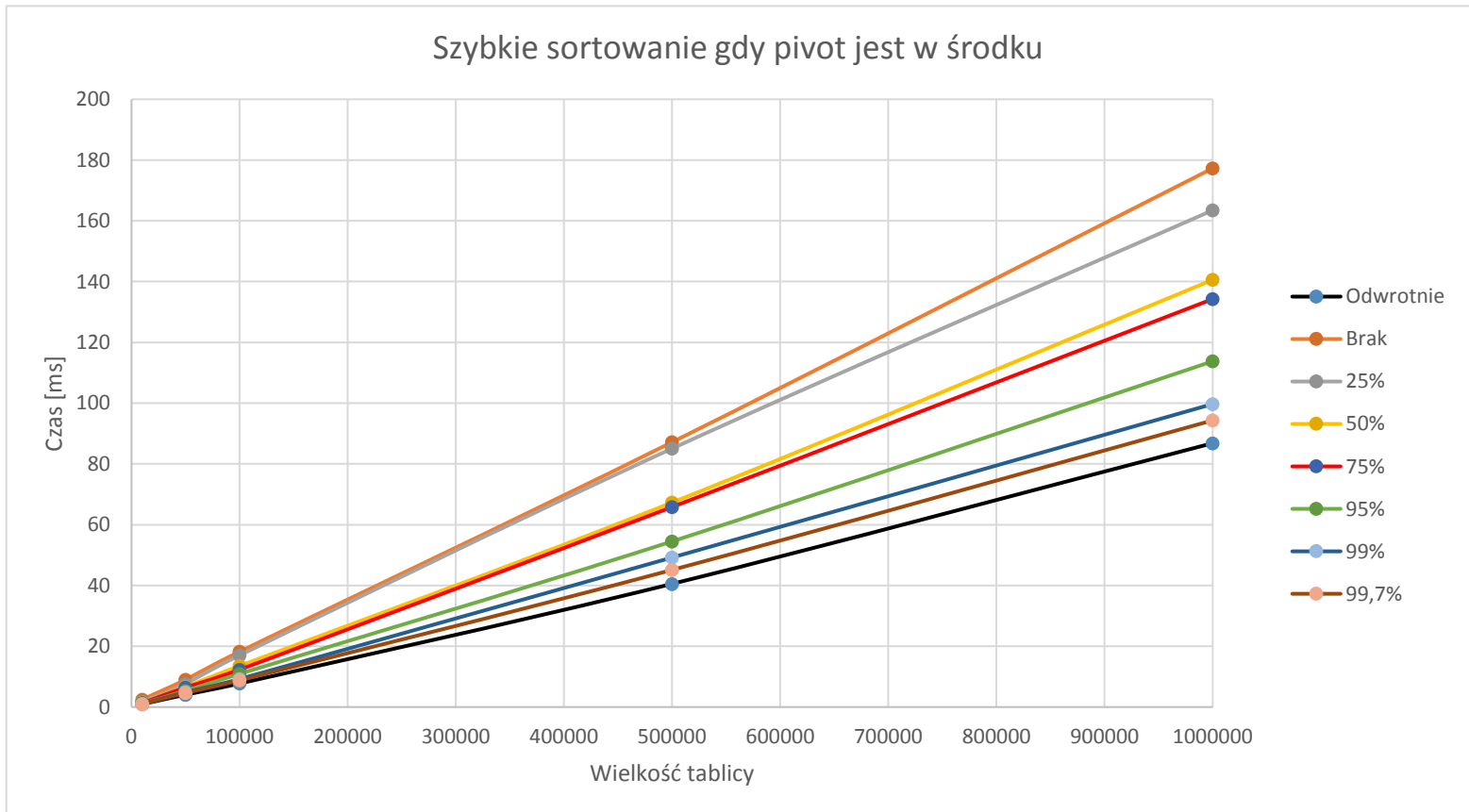
Wykres1) Badanie sortowania introspektywnego



Wykres2) Badanie sortowania przez scalanie



Wykres3) Badanie sortowania szybkiego z pivotem na końcu



Wykres4) Badanie sortowania szybkiego z pivotem w środku

Wnioski i podsumowanie

Sortowanie introspektywne cechowało się tym że najdłuższy czas sortowania był w przypadku tablicy która nie była posortowana, a najkrótszy gdy była posortowana odwrotnie. Można zauważyć że im bardziej wstępnie byłą posortowana tablica tym czas sortowania był krótszy.

Podobnie jak w przypadku sortowania introspektywnym najdłuższy czas sortowania był w przypadku braku wstępnego posortowania, podobne było też że im bardziej tablica była wstępnie posortowana tym czas sortowania malał. Sortowanie tablicy która była posortowana w 99% algorytm potrzebował najmniej czasu, a w przypadku odwrotnej kolejności tablicy czas był prawie taki sam jak dla tablicy posortowanej w 75%.

W przypadku sortowania szybkiego gdzie zostały przeprowadzone dwa badania, w pierwszym gdzie pivot został umieszczony na samym końcu, zdecydowanie najgorszym przypadkiem było sortowanie tablicy odwrotnie posortowanej. Z kolei najlepszym to przypadek braku wstępnego sortownia. W drugim przypadku najgorszym czasem charakteryzowało się sortowanie niepostartowanej tablicy, mimo tego czas był bardzo porównywalny do przypadku z pivotem na samym końcu. Jest to spowodowane tym że liczby były umieszczone w tablicy w sposób losowy, i w zależności jaki element został przypisany pivotowi, czas sortowania mógłby być różny. Najlepszą opcją jest gdy pivot jest równy mediany całego zbioru, dlatego sortowania tablicy odwrotnie posortowanej ma najlepszy czas, ponieważ mediana jest elementem środkowym, a tendencja jest podobna jak w przypadku wcześniejszych sortowań czyli im bardziej posortowana tablica tym czas jest krótszy.

Wracając do przypadku z pivotem na końcu, gdzie czas sortowania odwrotnie posortowanej tablicy był najdłuższy, w ceku eliminacji długiego czasu sortowania, powstało sortowanie introspektywne które eliminuję ilość rekurencji i zmniejsza czas sortowania w przypadkach pesymistycznych dla sortowania szybkiego.

Podsumowując w zależności z jakim przypadkiem mamy do czynienia, trzeba świadomie wybrać metodę sortowania, ponieważ w niektórych przypadkach może to znacznie wydłużyć czas procesu. Najszybszym sortowaniem jest sortowanie szybkie, pod warunkiem dobrego dobrania pivota, jeśli nie jest to możliwe to można użyć sortowania introspektywnego, lecz implementacja algorytmu jest bardzo złożona, z kolei sortowanie przez scalanie ma prostą implementacje lecz jest najgorszym sortowaniem w badaniu.

Źródła

https://pl.wikipedia.org/wiki/Sortowanie_szybkie

https://pl.wikipedia.org/wiki/Sortowanie_przez_scalanie

https://pl.wikipedia.org/wiki/Sortowanie_introspektywne

http://www.algorytm.edu.pl/algorytmy-maturalne/sortowanie-przez-scalanie.html

http://www.algorytm.edu.pl/algorytmy-maturalne/quick-sort.html